Distributed compression – The algorithmic-information-theoretical view

Marius Zimand

Towson University

Institut Henri Poincaré - Feb. 2, 2016

Distributed compression vs. centralized compression

Alice and Bob have correlated strings x, and respectively y, which they want to compress.

- Scenario 1 (Centralized compression): They collaborate and compress together.
- Scenario 2 (Distributed compression): They compress separately.

Questions:

- What are the possible compression rates in the two scenarios?
- Is there are a difference between the two scenarios?

Distributed compression vs. centralized compression

Alice and Bob have correlated strings x, and respectively y, which they want to compress.

- Scenario 1 (Centralized compression): They collaborate and compress together.
- Scenario 2 (Distributed compression): They compress separately.

Questions:

- What are the possible compression rates in the two scenarios?
- Is there are a difference between the two scenarios?

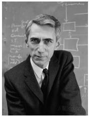
Answer: For quite general types of correlation, distributed compression can be on a par with centralized compression, provided the parties know how the data is correlated.

Modeling the correlation of x and y

- Statistical correlation:

  x and y are realizations of random variables X and Y;

  Their correlation is $H(X) + H(Y) - H(X, Y)$, where H is Shannon entropy.

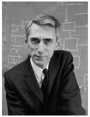


Modeling the correlation of x and y

- Statistical correlation:

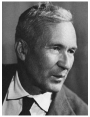
  x and y are realizations of random variables X and Y;

  Their correlation is $H(X) + H(Y) - H(X, Y)$, where H is Shannon entropy.



- Algorithmical correlation:

  Correlation of x and y: $C(x) + C(y) - C(x, y)$, where C is Kolmogorov complexity.



Information Theory view

- The n-bit strings x and y are realizations of r.v.'s X and Y.

Information Theory view

- The n-bit strings x and y are realizations of r.v.'s X and Y.
- X and Y are a 2-DMS (discrete memoryless source): n independent drawings from (X_1, Y_1), a joint distribution on bits .

Information Theory view

- The n-bit strings x and y are realizations of r.v.'s X and Y.
- X and Y are a 2-DMS (discrete memoryless source): n independent drawings from (X_1, Y_1), a joint distribution on bits .
- TASK: Alice uses compression function $E_1 : \{0,1\}^n \rightarrow \{1, 2, \ldots, 2^{nR_1}\}$.
  Bob uses compression function $E_2 : \{0,1\}^n \rightarrow \{1, 2, \ldots, 2^{nR_2}\}$.
  GOAL: (X, Y) can be reconstructed from the two encodings:
  There exists D such that with high probability: $D(E_1(X), E_2(Y)) = (X, Y)$.

Information Theory view

- The n-bit strings x and y are realizations of r.v.'s X and Y.
- X and Y are a 2-DMS (discrete memoryless source): n independent drawings from (X_1, Y_1), a joint distribution on bits .
- TASK: Alice uses compression function $E_1 : \{0,1\}^n \rightarrow \{1, 2, \ldots, 2^{nR_1}\}$.
  Bob uses compression function $E_2 : \{0,1\}^n \rightarrow \{1, 2, \ldots, 2^{nR_2}\}$.
  GOAL: (X, Y) can be reconstructed from the two encodings:
  There exists D such that with high probability: $D(E_1(X), E_2(Y)) = (X, Y)$.
- QUESTION: What compression rates R_1, R_2 can satisfy the task?

Information Theory view

- The n-bit strings x and y are realizations of r.v.'s X and Y.
- X and Y are a 2-DMS (discrete memoryless source): n independent drawings from (X_1, Y_1), a joint distribution on bits .
- TASK: Alice uses compression function $E_1 : \{0,1\}^n \rightarrow \{1, 2, \ldots, 2^{nR_1}\}$.
  Bob uses compression function $E_2 : \{0,1\}^n \rightarrow \{1, 2, \ldots, 2^{nR_2}\}$.
  GOAL: (X, Y) can be reconstructed from the two encodings:
  There exists D such that with high probability: $D(E_1(X), E_2(Y)) = (X, Y)$.
- QUESTION: What compression rates R_1, R_2 can satisfy the task?
- From Shannon Source Coding Theorem, it is necessary that

$$\begin{aligned} R_1 + R_2 &\geq H(X_1, Y_1) \\ R_1 &\geq H(X_1 \mid Y_1) \\ R_2 &\geq H(Y_1 \mid X_1). \end{aligned}$$

Slepian-Wolf Theorem

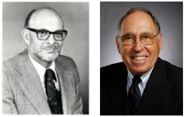
It is necessary that

$$R_1 + R_2 \geq H(X_1, Y_1), R_1 \geq H(X_1 \mid Y_1), R_2 \geq H(Y_1 \mid X_1).$$

Theorem (Slepian-Wolf)

Any R_1, R_2 satisfying strictly the above inequalities are sufficient for the task.

Slepian-Wolf Theorem



It is necessary that

$$R_1 + R_2 \geq H(X_1, Y_1), R_1 \geq H(X_1 \mid Y_1), R_2 \geq H(Y_1 \mid X_1).$$

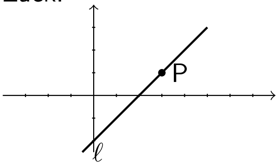
Theorem (Slepian-Wolf)

Any R_1, R_2 satisfying strictly the above inequalities are sufficient for the task.

- The theorem holds for any constant number of sources (not just two sources).
- The decompression procedure knows $H(X_1, X_2), H(X_1), H(X_2)$ – the information profile of the sources.
- The type of correlation is rather simple, because of the memoryless property.

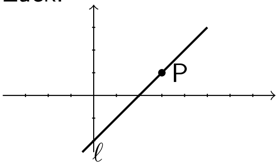
Algorithmic correlation: Motivating story

- Alice knows a line ℓ; Bob knows a point $P \in \ell$; They want to send ℓ and P to Zack.



Algorithmic correlation: Motivating story

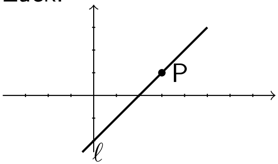
- Alice knows a line ℓ; Bob knows a point $P \in \ell$; They want to send ℓ and P to Zack.



- ℓ : $2n$ bits of information (intercept, slope in GF[2^n]).

Algorithmic correlation: Motivating story

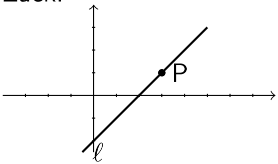
- Alice knows a line ℓ; Bob knows a point $P \in \ell$; They want to send ℓ and P to Zack.



- ℓ : $2n$ bits of information (intercept, slope in GF[2^n]).
- P : $2n$ bits of information (the 2 coord. in GF[2^n]).

Algorithmic correlation: Motivating story

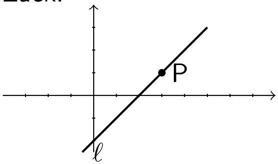
- Alice knows a line ℓ; Bob knows a point $P \in \ell$; They want to send ℓ and P to Zack.



- ℓ : $2n$ bits of information (intercept, slope in GF$[2^n]$).
- P : $2n$ bits of information (the 2 coord. in GF$[2^n]$).
- Total information in $(\ell, P) = 3n$ bits; mutual information of ℓ and $P = n$ bits.

Algorithmic correlation: Motivating story

- Alice knows a line ℓ; Bob knows a point $P \in \ell$; They want to send ℓ and P to Zack.

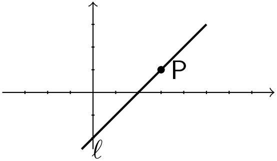


- ℓ : $2n$ bits of information (intercept, slope in GF$[2^n]$).
- P : $2n$ bits of information (the 2 coord. in GF$[2^n]$).
- Total information in $(\ell, P) = 3n$ bits; mutual information of ℓ and $P = n$ bits.
- QUESTION: Can Alice send $2n$ bits, and Bob n bits?

  Ans: Yes, of course. But is it just because of the simple geometric relation between ℓ and P?

Algorithmic correlation: Motivating story

- Alice knows a line ℓ; Bob knows a point $P \in \ell$; They want to send ℓ and P to Zack.



- ℓ : $2n$ bits of information (intercept, slope in GF[2^n]).
- P : $2n$ bits of information (the 2 coord. in GF[2^n]).
- Total information in $(\ell, P) = 3n$ bits; mutual information of ℓ and $P = n$ bits.
- QUESTION: Can Alice send $2n$ bits, and Bob n bits?

  Ans: Yes, of course. But is it just because of the simple geometric relation between ℓ and P?
- QUESTION: Can Alice send $1.5n$ bits, and Bob $1.5n$ bits? Can Alice send $1.74n$ bits, and Bob $1.26n$ bits?
  Ans: ???

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x)$ = length of a shortest program that produces x.

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x) =$ length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x) =$ length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;
- Assumptions:

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x)$ = length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;
- Assumptions:
  - $C(x) = 2n$; x has $2n$ bits of information.

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x) =$ length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;
- Assumptions:
  - $C(x) = 2n$; x has $2n$ bits of information.
  - $C(y) = 2n$; y has $2n$ bits of information.

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x)$ = length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;
- Assumptions:
  - $C(x) = 2n$; x has $2n$ bits of information.
  - $C(y) = 2n$; y has $2n$ bits of information.
  - $C(x, y) = 3n$ bits; so, the mutual information of x and y is n bits.

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x) =$ length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;
- Assumptions:
  - $C(x) = 2n$; x has $2n$ bits of information.
  - $C(y) = 2n$; y has $2n$ bits of information.
  - $C(x, y) = 3n$ bits; so, the mutual information of x and y is n bits.

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x)$ = length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;
- Assumptions:
  - $C(x) = 2n$; x has $2n$ bits of information.
  - $C(y) = 2n$; y has $2n$ bits of information.
  - $C(x, y) = 3n$ bits; so, the mutual information of x and y is n bits.
- QUESTION: How many bits do Alice and Bob need to send?

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x) =$ length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;
- Assumptions:
  - $C(x) = 2n$; x has $2n$ bits of information.
  - $C(y) = 2n$; y has $2n$ bits of information.
  - $C(x, y) = 3n$ bits; so, the mutual information of x and y is n bits.
- QUESTION: How many bits do Alice and Bob need to send?
  - Each one has to send at least n bits, and together at least $3n$ bits.

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x) =$ length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;
- Assumptions:
  - $C(x) = 2n$; x has $2n$ bits of information.
  - $C(y) = 2n$; y has $2n$ bits of information.
  - $C(x, y) = 3n$ bits; so, the mutual information of x and y is n bits.
- QUESTION: How many bits do Alice and Bob need to send?
  - Each one has to send at least n bits, and together at least $3n$ bits.
  - Can Alice send $2n$ bits, and Bob n bits? (Asymmetric Slepian-Wolf coding)

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x) =$ length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;
- Assumptions:
  - $C(x) = 2n$; x has $2n$ bits of information.
  - $C(y) = 2n$; y has $2n$ bits of information.
  - $C(x, y) = 3n$ bits; so, the mutual information of x and y is n bits.
- QUESTION: How many bits do Alice and Bob need to send?
  - Each one has to send at least n bits, and together at least $3n$ bits.
  - Can Alice send $2n$ bits, and Bob n bits? (Asymmetric Slepian-Wolf coding)
  - Can Alice send $1.5n$ bits, and Bob $1.5n$ bits?

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x) =$ length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;
- Assumptions:
  - $C(x) = 2n$; x has $2n$ bits of information.
  - $C(y) = 2n$; y has $2n$ bits of information.
  - $C(x, y) = 3n$ bits; so, the mutual information of x and y is n bits.
- QUESTION: How many bits do Alice and Bob need to send?
  - Each one has to send at least n bits, and together at least $3n$ bits.
  - Can Alice send $2n$ bits, and Bob n bits? (Asymmetric Slepian-Wolf coding)
  - Can Alice send $1.5n$ bits, and Bob $1.5n$ bits?
  - Can Alice send $1.74n$ bits, and Bob 1.26 bits?

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x)$ = length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;
- Assumptions:
  - $C(x) = 2n$; x has $2n$ bits of information.
  - $C(y) = 2n$; y has $2n$ bits of information.
  - $C(x, y) = 3n$ bits; so, the mutual information of x and y is n bits.
- QUESTION: How many bits do Alice and Bob need to send?
  - Each one has to send at least n bits, and together at least $3n$ bits.
  - Can Alice send $2n$ bits, and Bob n bits? (Asymmetric Slepian-Wolf coding)
  - Can Alice send $1.5n$ bits, and Bob $1.5n$ bits?
  - Can Alice send $1.74n$ bits, and Bob 1.26 bits?

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x)$ = length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;
- Assumptions:
  - $C(x) = 2n$; x has $2n$ bits of information.
  - $C(y) = 2n$; y has $2n$ bits of information.
  - $C(x, y) = 3n$ bits; so, the mutual information of x and y is n bits.
- QUESTION: How many bits do Alice and Bob need to send?
  - Each one has to send at least n bits, and together at least $3n$ bits.
  - Can Alice send $2n$ bits, and Bob n bits? (Asymmetric Slepian-Wolf coding)
  - Can Alice send $1.5n$ bits, and Bob $1.5n$ bits?
  - Can Alice send $1.74n$ bits, and Bob 1.26 bits?
- ANSWER: Yes, if Zack knows the complexity profile of x and y (and if we ignore logarithmic overhead).

Algorithmic correlation: Kolmogorov complexity

- Kolmogorov complexity:
  $C(x) =$ length of a shortest program that produces x.
- Alice knows x; Bob knows y; They want to send x, y to Zack;
- Assumptions:
  - $C(x) = 2n$; x has $2n$ bits of information.
  - $C(y) = 2n$; y has $2n$ bits of information.
  - $C(x, y) = 3n$ bits; so, the mutual information of x and y is n bits.
- QUESTION: How many bits do Alice and Bob need to send?
  - Each one has to send at least n bits, and together at least $3n$ bits.
  - Can Alice send $2n$ bits, and Bob n bits? (Asymmetric Slepian-Wolf coding)
  - Can Alice send $1.5n$ bits, and Bob $1.5n$ bits?
  - Can Alice send $1.74n$ bits, and Bob 1.26 bits?
- ANSWER: Yes, if Zack knows the complexity profile of x and y (and if we ignore logarithmic overhead).
- The main focus of this talk is to explain this answer.

Muchnik's Theorem (1)

- Alice has x, Bob has y.
- There is a string p of length $C(x \mid y)$ such that (p, y) is a program for x.
- p can be found from x, y with $\log n$ help bits.
- Can Alice alone compute p?
- In absolute terms, the answer is NO.
- **Muchnik's Theorem.** Using a few help bits and with a small overhead in the length, the answer is YES.

Muchnik's Theorem (2)

Theorem (Muchnik's Theorem)

For every x, y of complexity at most n, there exists p such that

- $|p| = C(x \mid y) + O(\log n)$.
- $C(p \mid x) = O(\log n)$
- $C(x \mid p, y) = O(\log\ n)$

Muchnik's Theorem (2)

overhead

Theorem (Muchnik's Theorem)

For every x, y of complexity at most n, there exists p such that

- $|p| = C(x \mid y) + O(\log n)$.
- $C(p \mid x) = O(\log n)$
- $C(x \mid p, y) = O(\log\ n)$

Muchnik's Theorem (2)

help bits

Theorem (Muchnik's Theorem)

For every x, y of complexity at most n, there exists p such that

- $|p| = C(x \mid y) + O(\log n)$.
- $C(p \mid x) = O(\log n)$
- $C(x \mid p, y) = O(\log n)$

help bits

Polynomial-time version of Muchnik's Th.

Theorem (Bauwens, Makhlin, Vereshchagin, Z., 2013)

For every x, y of complexity at most n, there exists p such that

- $|p| = C(x \mid y) + O(\log n)$.
- $C^{\text{poly}}(p \mid x) = O(\log n)$
- $C(x \mid p, y) = O(\log n)$

Theorem (Teutsch, 2014)

For every x, y of complexity at most n, there exists p such that

- $|p| = C(x \mid y) + O(1)$.
- $C^{\text{poly}}(p \mid x) = O(\log n)$
- (p, y) *is a program for* x.

Asymmetric Slepian-Wolf with help bits

- Alice knows x; Bob knows y. Suppose $C(x) = 2n$, $C(y) = 2n$, $C(x, y) = 3n$.
- QUESTION: Can Alice communicate x to *Bob* by sending him n bits?
- Muchnik's Theorem: Since $C(x \mid y) \approx n$, Alice, using only $O(\log n)$ help bits, can compute in polynomial time a string p of length $\approx n$, such that Bob can reconstruct x from (p, y).

Kolmogorov complexity version of the Slepian-Wolf Th. with help bits

Theorem (Romashchenko, 2005)

Let x, y be n-bit strings and s, t numbers such that

- $s + t \geq C(x, y)$
- $s \geq C(x \mid y)$
- $t \geq C(y \mid x)$.

There exists strings p, q such that

(1) $|p| = s + O(\log n)$, $|q| = t + O(\log n)$.

(2) $C(p \mid x) = O(\log n)$, $C(q \mid y) = O(\log n)$

(3) (p, q) *is a program for* (x, y).

Note: Romashchenko's theorem holds for an arbitrary constant number of sources.

- Alice knows x; Bob knows y. Suppose $C(x) = 2n, C(y) = 2n, C(x, y) = 3n$.
- QUESTION: Can Alice and Bob communicate x, y to *Zack*, each one sending $3n/2$ bits (or $1.74n$, respectively $1.26n$)?
- Romashchenko's theorem: YES (modulo the $O(\log n)$ overhead), provided they have a few help bits.

- QUESTION: Can we get rid of the help bits?
- Effective compression at minimum description length is impossible, so the compression/decompression procedures must have some additional information.
- But maybe we can replace the arbitrary help bits with some meaningful information which is more likely to be available in applications.
- Recall the example when Alice knows the line ℓ and Bob knows a point P. It may be that Alice, Bob and Zack know that the data is correlated in this way: $P \in \ell$. Can they take advantage of this?

An easier problem: single source compression

- Alice knows x and $C(x)$; then she can find a shortest program for x by exhaustive search.
- The running time is larger than any computable function.

Theorem (Bauwens, Z., 2014)

Let $t(n)$ be a computable function. If an algorithm on input $(x, C(x))$ computes in time $t(n)$ a program p for x, then $|p| = C(x) + \Omega(n)$ (where $n = |x|$).

An easier problem: single source compression

- Alice knows x and $C(x)$; then she can find a shortest program for x by exhaustive search.
- The running time is larger than any computable function.

Theorem (Bauwens, Z., 2014)

Let $t(n)$ be a computable function. If an algorithm on input $(x, C(x))$ computes in time $t(n)$ a program p for x, then $|p| = C(x) + \Omega(n)$ (where $n = |x|$).

Theorem (Bauwens, Z., 2014)

There exists algorithms E and D such that E runs in probabilistic poly. time and for all n-bit strings x, for all $\epsilon > 0$,

1. *E on input $x, C(x)$ and $1/\epsilon$, outputs a string p of length $\leq C(x) + \log^2(n/\epsilon)$,*
2. *D on input $p, C(x)$ outputs x with probability $1 - \epsilon$.*

- So, finding a short program for x, given x and $C(x)$, can be done in probabilistic poly. time, but any deterministic algorithm takes time larger than any computable function.
- The decompressor D cannot run in polynomial time, when compression is done at minimum description length (or close to it).

Kolmogorov complexity version of the Slepian-Wolf Th. - asymmetric version

Theorem (Bauwens, Z, 2014)

There exists a probabilistic poly. time algorithm A such that

- *On input* (x, ϵ) *and "promise" parameter* k, A *outputs* p,
- $|p| = k + O(\log^2(|x|/\epsilon))$,
- *If the "promise"* $k = C(x \mid y)$ *holds, then,*

with probability $(1 - \epsilon)$, (p, y) *is a program for* x.

Kolmogorov complexity version of the Slepian-Wolf Th. - asymmetric version

Theorem (Bauwens, Z, 2014)

There exists a probabilistic poly. time algorithm A such that

- *On input* (x, ϵ) *and "promise" parameter* k*, A outputs* p*,*
- $|p| = k + O(\log^2(|x|/\epsilon))$*,*
- *If the "promise"* $k = C(x \mid y)$ *holds, then,*

with probability $(1 - \epsilon)$*,* (p, y) *is a program for* x*.*

- Alice has x, Bob has y; they want to send x, y to Zack.
- Suppose: $C(x) = 2n, C(y) = 2n, C(x, y) = 3n$.
- Bob can send y, and Alice can compress x to p of length $n + \log^2 n$, provided she knows $C(x \mid y)$.

Kolmogorov complexity version of the Slepian-Wolf Theorem- 2 sources

Theorem (Z., 2015)

There exist probabilistic poly.-time algorithms E_1, E_2 and algorithm D such that for all integers n_1, n_2 and n-bit strings x_1, x_2,

if $n_1 + n_2 \geq C(x_1, x_2)$, $n_1 \geq C(x_1 \mid x_2)$, $n_2 \geq C(x_2 \mid x_1)$,

then

- *E_i on input (x_i, n_i) outputs a string p_i of length $n_i + O(\log^3 n)$, for $i = 1, 2$,*
- *D on input (p_1, p_2) and the complexity profile of (x_1, x_2) outputs (x_1, x_2) with probability $1 - 1/n$.*

(The complexity profile of (x_1, x_2) is the tuple $(C(x_1), C(x_2), C(x_1, x_2))$).

Kolmogorov complexity version of the Slepian-Wolf Theorem- ℓ sources

- The case of ℓ senders: sender i has string x_i, $i \in [\ell]$.
- If $V = \{i_1, \ldots, i_k\}$, we denote $x_V = (x_{i_1}, \ldots, x_{i_k})$.
- The complexity profile of $(x_1, \ldots, x_\ell)$ is the set of integers $\{C(x_V) \mid V \subseteq [\ell]\}$.

Theorem (Z., 2015)

There exist probabilistic poly.-time algorithms $E_1, \ldots, E_\ell$, algorithm D and a function $\alpha(n) = \log^{O_\ell(1)}(n)$, such that for all integers $n_1, \ldots, n_\ell$ and n-bit strings $x_1, \ldots, x_\ell$,
if $\sum_{i \in V} n_i \geq C(x_V \mid x_{[\ell]-V})$, for all $V \subseteq [\ell]$,
then

- *E_i on input (x_i, n_i) outputs a string p_i of length $n_i + \alpha(n)$, for $i \in [\ell]$,*
- *D on input $(p_1, \ldots, p_\ell)$ and the complexity profile of $(x_1, \ldots, x_\ell)$ outputs $(x_1, \ldots, x_\ell)$ with probability $1 - 1/n$.*

On the promise conditions

- **[Real Theorem]** There exists a poly-time probabilistic algorithm A that on input (x, k) returns a string p of length $k + \mathrm{poly}(\log |x|)$ such that if $C(x \mid y) = k$, then with high probability (p, y) is a program for x.

- The promise condition:

  Alice knows $k = C(x \mid y)$.

On the promise conditions

- **[Real Theorem]** There exists a poly-time probabilistic algorithm A that on input (x, k) returns a string p of length $k + \mathrm{poly}(\log |x|)$ such that if $C(x \mid y) = k$, then with high probability (p, y) is a program for x.
- The promise condition:

  Alice knows $k = C(x \mid y)$.
- Can it be relaxed to
  Alice knows $k \geq C(x \mid y)$?
- **[Dream Theorem ???]** $\cdots$ if $C(x \mid y) \leq k$ $\cdots$.

On the promise conditions

- **[Real Theorem]** There exists a poly-time probabilistic algorithm A that on input (x, k) returns a string p of length $k + \mathrm{poly}(\log |x|)$ such that if $C(x \mid y) = k$, then with high probability (p, y) is a program for x.
- The promise condition:

  Alice knows $k = C(x \mid y)$.
- Can it be relaxed to
  Alice knows $k \geq C(x \mid y)$?
- **[Dream Theorem ???]** $\cdots$ if $C(x \mid y) \leq k$ $\cdots$.
- Dream Theorem open.

Weaker version of the Dream Theorem.

Theorem (Z)

Let us assume complexity assumption H holds.
Let q be some polynomial.
There exists a poly time, probabilistic algorithm A that on input (x, k) returns a string p of length $k + O(\log |x|/\epsilon)$ such that if $C^q(x \mid y) \leq k$, then, with probability $1 - \epsilon$, (p, y) is a program for x.

Weaker version of the Dream Theorem.

Theorem (Z)

Let us assume complexity assumption H holds.
Let q be some polynomial.
There exists a poly time, probabilistic algorithm A that on input (x, k) returns a string p of length $k + O(\log |x|/\epsilon)$ such that if $C^q(x \mid y) \leq k$, then, with probability $1 - \epsilon$, (p, y) is a program for x.

Assumption *H*

$\exists f \in \mathrm{E}$ which cannot be computed in space $2^{o(n)}$.

Weaker version of the Dream Theorem.

Theorem (Z)

Let us assume complexity assumption H holds.
Let q be some polynomial.
There exists a poly time, probabilistic algorithm A that on input (x, k) returns a string p of length $k + O(\log |x|/\epsilon)$ such that if $C^q(x \mid y) \leq k$, then, with probability $1 - \epsilon$, (p, y) is a program for x.

Assumption H

$\exists f \in \mathrm{E}$ which cannot be computed in space $2^{o(n)}$.

$\mathrm{E} = \cup_{c>0} \mathrm{DTIME}[2^{cn}]$

Some proof sketches...

First proof

Theorem (Bauwens, Z, 2014)

There exists a probabilistic poly. time algorithm A such that

- *On input* (x, δ) *and promise parameter* k, *A outputs* p,
- $|p| = k + \log^2(|x|/\delta)$,
- *If the promise condition* $k = C(x \mid y)$ *holds, then,*

with probability $(1 - \delta)$, (p, y) *is a program for* x.

First proof

Theorem (Bauwens, Z, 2014)

There exists a probabilistic poly. time algorithm A such that
- *On input* (x, δ) *and promise parameter* k, *A outputs* p,
- $|p| = k + \log^2(|x|/\delta)$,
- *If the promise condition* $k = C(x \mid y)$ *holds, then,*

with probability $(1 - \delta)$, (p, y) *is a program for* x.

To keep the notation simple, I will assume that y is the empty string, and I will drop y.
Essentially the same proof works for arbitrary y.

Combinatorial object

Key tool: bipartite graphs $G = (L, R, E \subseteq L \times R)$ with the rich owner property:

For any $B \subseteq L$ of size $|B| \approx K$, most x in B own most of their neighbors (these neighbors are not shared with any other node from B).

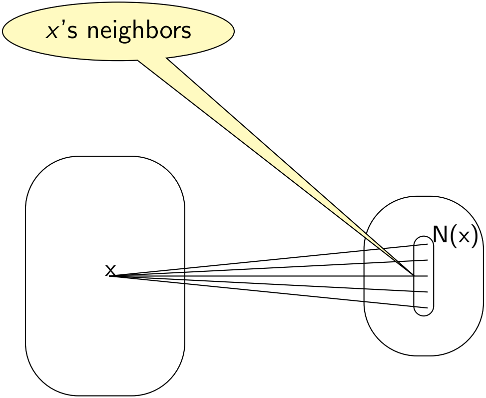
Combinatorial object

Key tool: bipartite graphs $G = (L, R, E \subseteq L \times R)$ with the rich owner property:

For any $B \subseteq L$ of size $|B| \approx K$, most x in B own most of their neighbors (these neighbors are not shared with any other node from B).

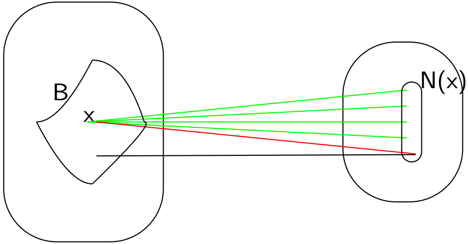
- $x \in B$ owns $y \in N(x)$ w.r.t. B if $N(y) \cap B = \{x\}$.
- $x \in B$ is a rich owner if x owns $(1-\delta)$ of its neighbors w.r.t. B.
- $G = (L, R, E \subseteq L \times R)$ has the (K, δ)-rich owner property if
for all B with $|B| \leq K$, $(1-\delta)K$ of the elements in B are rich owners w.r.t. B.

Bipartite graph G



Bipartite graph G

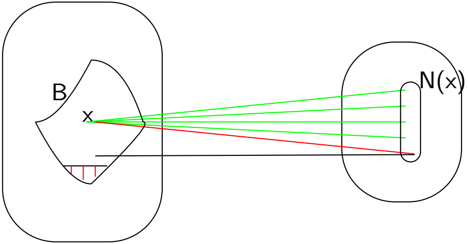
x is a rich owner
w.r.t B
if x owns $(1 - \delta)$ of
$N(x)$



Bipartite graph G

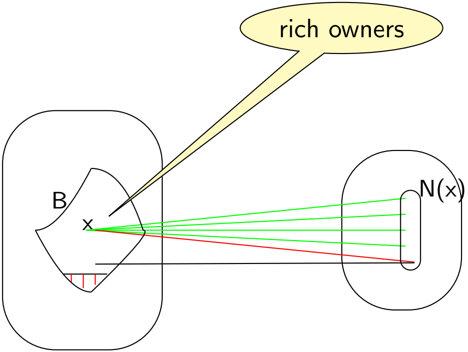
x is a rich owner w.r.t B if x owns $(1-\delta)$ of $N(x)$

G has the (K, δ) rich owner property: $\forall B \subseteq L$, of size at most K, all nodes in B except at most $\delta \cdot K$ are rich owners w.r.t. B



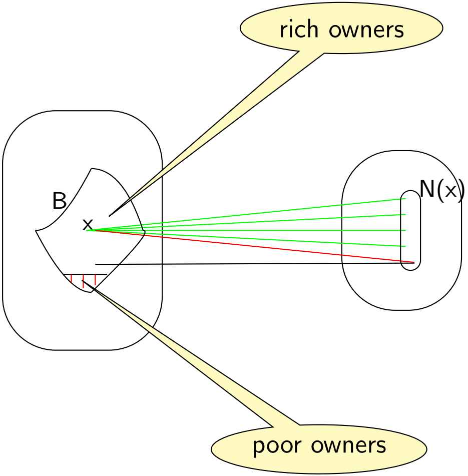
x is a rich owner w.r.t B
if x owns $(1-\delta)$ of $N(x)$

G has the (K, δ) rich owner property:
$\forall B \subseteq L$, of size at most K,
all nodes in B except at most $\delta \cdot K$ are rich owners w.r.t. B



x is a rich owner w.r.t B
if x owns $(1-\delta)$ of $N(x)$

G has the (K, δ) rich owner property:
$\forall B \subseteq L$, of size at most K,
all nodes in B except at most $\delta \cdot K$ are rich owners w.r.t. B

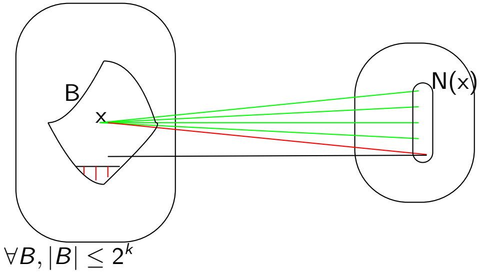


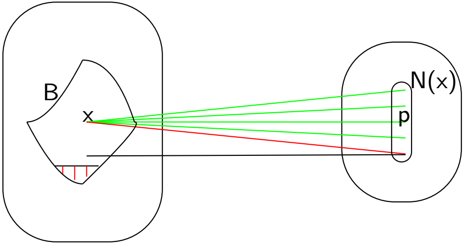
Theorem (Bauwens, Z'14)

There exists a computable (uniformly in n, k and $1/\delta$) graph with the rich owner property for parameters $(2^k, \delta)$ with:

- $L = \{0,1\}^n$
- $R = \{0,1\}^{k+O(\log(n/\delta))}$
- $D(\textit{left degree}) = \mathrm{poly}(n/\delta)$

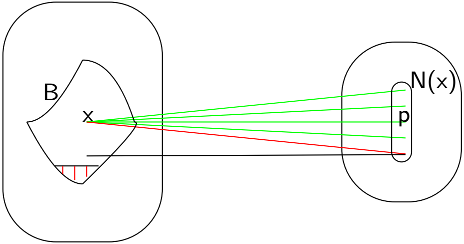
Similar for poly-time G, except overhead in R is $O(\log^2(n/\delta))$ and $D = 2^{O(\log^2(n/\delta))}$.





- Any $p \in N(x)$ owned by x w.r.t. $B = \{x' \mid C(x') \leq k\}$ is a program for x.

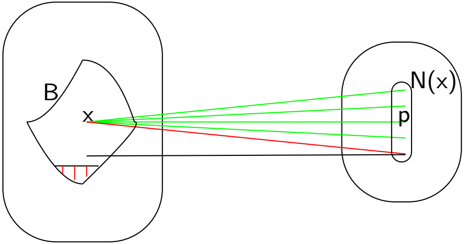
How to construct x from p: Enumerate B till we find an element that owns p. This is x.



- Any $p \in N(x)$ owned by x w.r.t. $B = \{x' \mid C(x') \leq k\}$ is a program for x.

How to construct x from p: Enumerate B till we find an element that owns p. This is x.

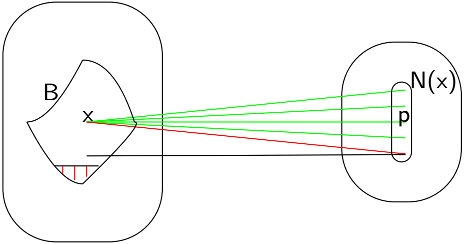
- So if x is a rich owner, $(1 - \delta)$ of his neighbors are programs for it.



- Any $p \in N(x)$ owned by x w.r.t. $B = \{x' \mid C(x') \leq k\}$ is a program for x.

How to construct x from p: Enumerate B till we find an element that owns p. This is x.

- So if x is a rich owner, $(1-\delta)$ of his neighbors are programs for it.
- What if x is a poor owner? There are few poor owners, so x has complexity $< k$.



- Any $p \in N(x)$ owned by x w.r.t. $B = \{x' \mid C(x') \leq k\}$ is a program for x.

How to construct x from p: Enumerate B till we find an element that owns p. This is x.

- So if x is a rich owner, $(1 - \delta)$ of his neighbors are programs for it.
- What if x is a poor owner? There are few poor owners, so x has complexity $< k$.
- So if $C(x) = k$, we compress x by picking at random one of its neighbors.

Building graphs with the rich owner property

- Step 1: $(1-\delta)$ of $x \in B$ **partially** own $(1-\delta)$ of its neighbors.

Building graphs with the rich owner property

shared with only $\mathrm{poly}(n)$ nodes

- Step 1: $(1-\delta)$ of $x \in B$ **partially** own $(1-\delta)$ of its neighbors.

Building graphs with the rich owner property

shared with only $\mathrm{poly}(n)$ nodes

- Step 1: $(1-\delta)$ of $x \in B$ **partially** own $(1-\delta)$ of its neighbors.
- Step 2: $(1-\delta)$ of $x \in B$ ~~**partially**~~ own $(1-\delta)$ of its neighbors.

Building graphs with the rich owner property

shared with only $\mathrm{poly}(n)$ nodes

- Step 1: $(1-\delta)$ of $x \in B$ **partially** own $(1-\delta)$ of its neighbors.

- Step 2: $(1-\delta)$ of $x \in B$ ~~**partially**~~ own $(1-\delta)$ of its neighbors.

Step 1 is done with extractors that have small entropy loss.

Step 2 is done by hashing.

Extractors

$E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ϵ)-extractor if for any $B \subseteq \{0,1\}^n$ of size $|B| \geq 2^k$ and for any $A \subseteq \{0,1\}^m$,

$$|\mathrm{Prob}(E(U_B, U_d) \in A) - \mathrm{Prob}(U_m \in A)| \leq \epsilon,$$

Extractors

uniform distr. on $\{0,1\}^m$

$E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ϵ)-extractor for any $B \subseteq \{0,1\}^n$ of size $|B| \geq 2^k$ and for any $A \subseteq \{0,1\}^m$,

$$|\mathrm{Prob}(E(U_B, U_d) \in A) - \mathrm{Prob}(U_m \in A)| \leq \epsilon,$$

uniform distr. on B

uniform distr. on $\{0,1\}^d$

Extractors

$E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ)-extractor if for any $B \subseteq \{0,1\}^n$ of size $|B| \geq 2^k$ and for any $A \subseteq \{0,1\}^m$,

$$|\mathrm{Prob}(E(U_B, U_d) \in A) - \mathrm{Prob}(U_m \in A)| \leq \epsilon,$$

or, in other words,

$$\left| \frac{|E(B,A)|}{|B| \cdot 2^d} - \frac{|A|}{2^m} \right| \leq \epsilon.$$

The entropy loss is $s = k + d - m$.

Step 1

GOAL : $\forall B \subseteq L$ with $|B| \approx K$, most nodes in B share most of their neighbors with only $\mathrm{poly}(n)$ other nodes from B.

We can view an extractor E as a bipartite graph G_E with $L = \{0,1\}^n, R = \{0,1\}^m$ and left-degree $D = 2^d$.

If E is a (k, ϵ)-extractor, then it has low congestion:
for any $B \subseteq L$ of size $|B| \approx 2^k$, most $x \in B$ share most of their neighbors with only $O(1/\epsilon \cdot 2^s)$ other nodes in B.

Step 1

GOAL : $\forall B \subseteq L$ **with** $|B| \approx K$**, most nodes in** B **share most of their neighbors with only** $\mathrm{poly}(n)$ **other nodes from** B**.**

We can view an extractor E as a bipartite graph G_E with $L = \{0,1\}^n, R = \{0,1\}^m$ and left-degree $D = 2^d$.

If E is a (k, ϵ)-extractor, then it has low congestion: proof on next slide
for any $B \subseteq L$ of size $|B| \approx 2^k$, most $x \in B$ share most of their neighbors with only $O(1/\epsilon \cdot 2^s)$ other nodes in B.

Step 1

GOAL : $\forall B \subseteq L$ with $|B| \approx K$, most nodes in B share most of their neighbors with only $\mathrm{poly}(n)$ other nodes from B.

We can view an extractor E as a bipartite graph G_E with $L = \{0,1\}^n, R = \{0,1\}^m$ and left-degree $D = 2^d$.

If E is a (k, ϵ)-extractor, then it has low congestion:
for any $B \subseteq L$ of size $|B| \approx 2^k$, most $x \in B$ share most of their neighbors with only $O(1/\epsilon \cdot 2^s)$ other nodes in B.

By the probabilistic method: There are extractors whith entropy loss $s = O(\log(1/\epsilon))$ and log-left degree $d = O(\log n/\epsilon)$.

[Guruswami, Umans, Vadhan, 2009] Poly-time extractors with entropy loss $s = O(\log(1/\epsilon))$ and log-left degree $d = O(\log^2 n/\epsilon)$.

So for $1/\epsilon = \mathrm{poly}(n)$, we get our GOAL.

Extractors have low congestion

DEF: $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ)-extractor if for any $B \subseteq \{0,1\}^n$ of size $|B| \geq 2^k$ and for any $A \subseteq \{0,1\}^m$, $|\mathrm{Prob}(E(U_B, U_d) \in A) - \mathrm{Prob}(A)| \leq \epsilon$.
The entropy loss is $s = k + d - m$.

Extractors have low congestion

DEF: $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ϵ)-extractor if for any $B \subseteq \{0,1\}^n$ of size $|B| \geq 2^k$ and for any $A \subseteq \{0,1\}^m$, $|\mathrm{Prob}(E(U_B, U_d) \in A) - \mathrm{Prob}(A)| \leq \epsilon$.
The entropy loss is $s = k + d - m$.

Lemma

*Let E be a (k,ϵ)-extractor, $B \subseteq L$, $|B| = \frac{1}{\epsilon}2^k$.
Then all $x \in B$, except at most 2^k, share $(1-2\epsilon)$ of $N(x)$ with at most $2^s(\frac{1}{\epsilon})^2$ other nodes in B.*

Extractors have low congestion

DEF: $E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ)-extractor if for any $B \subseteq \{0,1\}^n$ of size $|B| \geq 2^k$ and for any $A \subseteq \{0,1\}^m$, $|\mathrm{Prob}(E(U_B, U_d) \in A) - \mathrm{Prob}(A)| \leq \epsilon$.
The entropy loss is $s = k + d - m$.

Lemma

Let E be a (k, ϵ)-extractor, $B \subseteq L$, $|B| = \frac{1}{\epsilon}2^k$.
Then all $x \in B$, except at most 2^k, share $(1 - 2\epsilon)$ of $N(x)$ with at most $2^s(\frac{1}{\epsilon})^2$ other nodes in B.

PROOF. Restrict left side to B. Avg-right-degree $= \frac{|B|2^d}{2^m} = \frac{1}{\epsilon} \cdot 2^s$.

Take A - the set of right nodes with $\deg_B \geq (2^s(1/\epsilon)) \cdot (1/\epsilon)$. Then $|A|/|R| \leq \epsilon$.

Take B' the nodes in B that do not have the property, i.e., they have $> 2\epsilon$ fraction of neighbors in A.

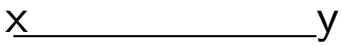
$|\mathrm{Prob}(E(U_{B'}, U_d) \in A) - |A|/|R|| > |2\epsilon - \epsilon| = \epsilon$.

So $|B'| \leq 2^k$.

Step 2

GOAL: Reduce sharing most neighbors with $\mathrm{poly}(n)$ other nodes, to sharing them with no other nodes.

y is shared by x with $x_2, \ldots, x_{\mathrm{poly}(n)}$



Step 2

GOAL: Reduce sharing most neighbors with $\mathrm{poly}(n)$ other nodes, to sharing them with no other nodes.

Let $x_1, x_2, \ldots, x_{\mathrm{poly}(n)}$ be n-bit strings.

Consider $p_1, \ldots, p_T$ the first T prime numbers, where $T = (1/\delta) \cdot n \cdot \mathrm{poly}(n)$.

For every x_i, for $(1 - \delta)$ of the T prime numbers, $(x_i \bmod p)$ is unique in $(x_1 \bmod p, \ldots, x_T \bmod p)$.

y is shared by x with $x_2, \ldots, x_{\mathrm{poly}(n)}$

x ____________ y

Step 2

GOAL: Reduce sharing most neighbors with $\mathrm{poly}(n)$ other nodes, to sharing them with no other nodes.

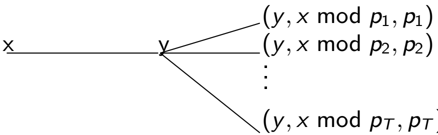
Let $x_1, x_2, \ldots, x_{\mathrm{poly}(n)}$ be n-bit strings.

Consider $p_1, \ldots, p_T$ the first T prime numbers, where $T = (1/\delta) \cdot n \cdot \mathrm{poly}(n)$.

For every x_i, for $(1-\delta)$ of the T prime numbers, $(x_i \bmod p)$ is unique in $(x_1 \bmod p, \ldots, x_T \bmod p)$.

In this way, by "splitting" each edge into T new edges we reach our GOAL.

y is shared by x with $x_2, \ldots, x_{\mathrm{poly}(n)}$



Step 2

GOAL: Reduce sharing most neighbors with $\mathrm{poly}(n)$ other nodes, to sharing them with no other nodes.

Let $x_1, x_2, \ldots, x_{\mathrm{poly}(n)}$ be n-bit strings.

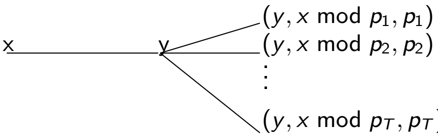
Consider $p_1, \ldots, p_T$ the first T prime numbers, where $T = (1/\delta) \cdot n \cdot \mathrm{poly}(n)$.

For every x_i, for $(1-\delta)$ of the T prime numbers, (x_i mod p) is unique in (x_1 mod $p, \ldots, x_T$ mod p).

In this way, by "splitting" each edge into T new edges we reach our GOAL.

Cost: overhead of $O(\log n)$ to the right nodes and the left degree increases by a factor of $T = \mathrm{poly}(n)$.

y is shared by x with $x_2, \ldots, x_{\mathrm{poly}(n)}$



2-nd proof: Kolmogorov complexity version of the Slepian-Wolf Theorem- 2 sources

Theorem (Z,2015)

There exist probabilistic poly.-time algorithms E_1, E_2 and algorithm D such that for all integers n_1, n_2 and n-bit strings x_1, x_2,

if $n_1 + n_2 \geq C(x_1, x_2)$, $n_1 \geq C(x_1 \mid x_2)$, $n_2 \geq C(x_2 \mid x_1)$,

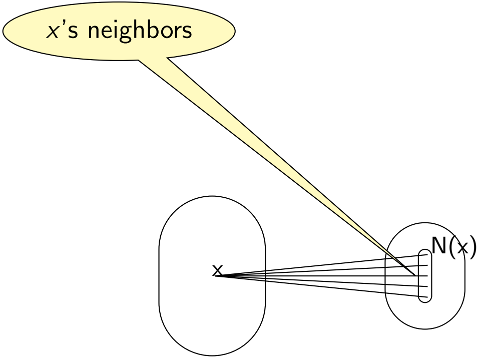
then

- *E_i on input (x_i, n_i) outputs a string p_i of length $n_i + O(\log^3 n)$, for $i = 1, 2$,*
- *D on input (p_1, p_2) and the complexity profile of (x_1, x_2) outputs (x_1, x_2) with probability $1 - 1/n$.*

(The complexity profile of (x_1, x_2) is the tuple $(C(x_1), C(x_2), C(x_1, x_2))$).

Graphs with the rich owner property - extended version

Bipartite graph G, with left degree D; parameters k, δ;



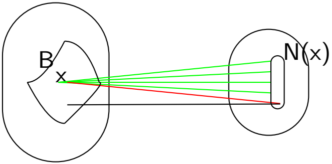
Graphs with the rich owner property - extended version

Bipartite graph G, with left degree D; parameters k, δ;

x is a rich owner w.r.t B if

small regime case: $|B| \leq 2^k$
x owns $(1-\delta)$ of $N(x)$

large regime case: $|B| \geq 2^k$
at least fraction $(1-\delta)$ of $y \in N(x)$ have $\deg_B(y) \leq (2/\delta^2)|B|D/2^k$



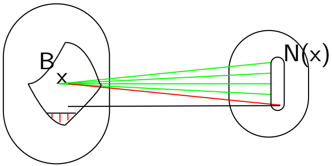
Graphs with the rich owner property - extended version

Bipartite graph G, with left degree D; parameters k, δ;

x is a rich owner w.r.t B if

small regime case: $|B| \leq 2^k$
x owns $(1-\delta)$ of $N(x)$

large regime case: $|B| \geq 2^k$
at least fraction $(1-\delta)$ of $y \in N(x)$ have $\deg_B(y) \leq (2/\delta^2)|B|D/2^k$



G has the (k, δ) rich owner property:
$\forall B \subseteq L$,
all nodes in B except at most $\delta \cdot |B|$ are rich owners w.r.t. B

Graphs with the rich owner property - extended version

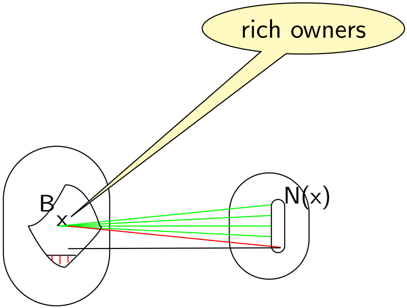
Bipartite graph G, with left degree D; parameters k, δ;

x is a rich owner w.r.t B if

small regime case: $|B| \leq 2^k$
x owns $(1-\delta)$ of $N(x)$

large regime case: $|B| \geq 2^k$
at least fraction $(1-\delta)$ of $y \in N(x)$ have $\deg_B(y) \leq (2/\delta^2)|B|D/2^k$

G has the (k, δ) rich owner property:
$\forall B \subseteq L$,
all nodes in B except at most $\delta \cdot |B|$ are rich owners w.r.t. B



Graphs with the rich owner property - extended version

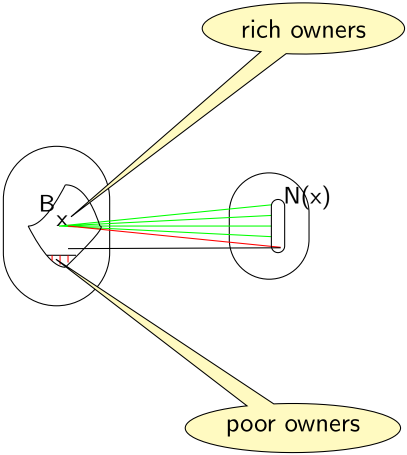
Bipartite graph G, with left degree D; parameters k, δ;

x is a rich owner w.r.t B if

small regime case: $|B| \leq 2^k$
x owns $(1-\delta)$ of $N(x)$

large regime case: $|B| \geq 2^k$
at least fraction $(1-\delta)$ of $y \in N(x)$ have $\deg_B(y) \leq (2/\delta^2)|B|D/2^k$

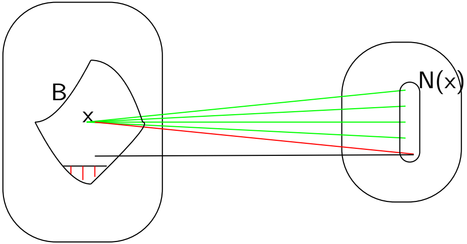
G has the (k, δ) rich owner property:
$\forall B \subseteq L$,
all nodes in B except at most $\delta \cdot |B|$ are rich owners w.r.t. B



Theorem

There exists a poly.-time computable (uniformly in n, k and $1/\delta$) graph with the rich owner property for parameters (k, δ) with:

- $L = \{0,1\}^n$
- $R = \{0,1\}^{k+O(\log^3(n/\delta))}$
- $D(\textit{left degree}) = 2^{O(\log^3(n/\delta))}$



Proof sketch

- Alice has x_1, Bob has x_2.
- They want to compress to lengths $n_1 + O(\log^3(n/\delta))$, resp. $n_2 + O(\log^3(n/\delta))$.
- Hypothesis: $n_1 \geq C(x_1 \mid x_2), n_2 \geq C(x_2 \mid x_1), n_1 + n_2 \geq C(x_1, x_2)$.
- Alice uses G_1, graph with the (n_1, δ) rich owner property. She compresses by choosing p_1, a random neighbor of x_1 in G_1.
- Bob uses G_2, graph with the (n_2, δ) rich owner property. He compresses by choosing p_2, a random neighbor of x_2 in G_2.
- Receiver reconstructs x_1, x_2 from p_1, p_2.

Reconstruction of x_1, x_2 from p_1, p_2

Case 1: $C(x_2) \leq n_2$.

- Let $B = \{x \mid C(x) \leq C(x_2)\}$.
- $|B| \leq 2^{C(x_2)} \leq 2^{n_2}$. So, B is in the small regime in G_2.
- Claim: x_2 can be reconstructed from p_2 by the following argument.
  - $|\text{set of poor owners}| \leq \delta|B|$. So, poor owners have complexity $< C(x_2)$.
  - So, x_2 is a rich owner; with prob. $1 - \delta$, x_2 owns p_2 with respect to B.
  - x_2 can be reconstructed from p_2, by enumerating B till we see a neighbor of p_2.
- Next, let $B = \{x_1' \mid C(x_1' \mid x_2) \leq C(x_1 \mid x_2)\}$.
- $|B| \leq 2^{C(x_1 \mid x_2)} \leq 2^{n_1}$. So B is in the small regime in G_1.
- Using argument, x_1 can be reconstructed from p_1.

Reconstruction of x_1, x_2 from p_1, p_2 – (2)

Case 2: $C(x_2) > n_2$.

- **Claim 1.** $C(p_2) =^* n_2$ ($*$ means that we ignore polylog terms).
- Pf. Let $B = \{x \mid C(x) \leq C(x_2)\}$. B is in the large regime in G_2.
- With prob. $1 - \delta$, x_2 shares p_2 with at most $(2/\delta^2)|B|D/2^{n_2} = 2^{C(x_2) - n_2 + \text{polylogn}}$ other nodes in B.
- x_2 can be reconstructed from p_2 and its rank among p_2's neighbors in B.
- So, $C(x_2) \leq^* C(p_2) + (C(x_2) - n_2)$.
- So, $C(p_2) \geq^* n_2$. Since $|p_2| =^* n_2$, we get $C(p_2) =^* n_2$.

Reconstruction of x_1, x_2 from p_1, p_2 – (3)

- **Claim 2.** Given p_2, x_1 and $C(x_2 \mid x_1)$, receiver can reconstruct x_2
- Pf. $B = \{x_2' \mid C(x_2' \mid x_1) \leq C(x_2 \mid x_1)\}$ is in the small regime case, and we can use the argument.
- So, $C(x_2, x_1) \leq^* C(p_2, x_1)$.
- But $C(p_2, x_1) \leq^* C(x_2, x_1)$ (because p_2 can be obtained from x_2 and its rank among x_2's neighbors).
- So, $C(x_2, x_1) =^* C(p_2, x_1)$.
- So, $C(x_1 \mid p_2) =^* C(x_1, p_2) - C(p_2) =^* C(x_1, x_2) - n_2$.

Reconstruction of x_1, x_2 from p_1, p_2 – (4)

- **Claim 3.** x_1 can be reconstructed from p_1 and p_2. (So, by Claim 2, x_2 can also be reconstructed, and we are done.)
- Pf. $B = \{x_1' \mid C(x_1' \mid p_2) \leq C(x_1, x_2) - n_2\}$.
- $x_1 \in B$, by the previous equality.
- Since $C(x_1, x_2) - n_2 \leq (n_1 + n_2) - n_2 = n_1$, B is in the small regime case.
- Conclusion follows by argument.

Third proof.

Theorem (Z)

Let us assume complexity assumption H holds.
Let q be some polynomial.
There exists a poly time, probabilistic algorithm A that on input (x, k) returns a string p of length $k + O(\log |x|/\delta)$ such that if $C^q(x \mid y) \leq k$, then, with probability $1 - \delta$, (p, y) is a program for x.

Third proof.

Theorem (Z)

Let us assume complexity assumption H holds.
Let q be some polynomial.
There exists a poly time, probabilistic algorithm A that on input (x, k) returns a string p of length $k + O(\log |x|/\delta)$ such that if $C^q(x \mid y) \leq k$, then, with probability $1 - \delta$, (p, y) is a program for x.

Assumption H

$\exists f \in \mathrm{E}$ which cannot be computed in space $2^{o(n)}$.

Third proof.

Theorem (Z)

Let us assume complexity assumption H holds.
Let q be some polynomial.
There exists a poly time, probabilistic algorithm A that on input (x, k) *returns a string p of length* $k + O(\log |x|/\delta)$ *such that if* $C^q(x \mid y) \leq k$*, then, with probability* $1 - \delta$, (p, y) *is a program for x.*

Assumption *H*

$\exists f \in \mathrm{E}$ which cannot be computed in space $2^{o(n)}$.

$\mathrm{E} = \cup_{c>0}\mathrm{DTIME}[2^{cn}]$

Assumption H implies pseudo-random generators that fool PSPACE predicates

[Nisan-Wigderson'94, Klivans - van Melkebeek'02, Miltersen'01]

If H is true, then there exists a pseudo-random generator g that fools any predicate computable in PSPACE with polynomial advice.

There exists $g : \{0,1\}^{c \log n} \rightarrow \{0,1\}^n$ such that for any T computable in PSPACE with poly advice,

$$\left|\mathrm{Prob}[T(g(U_s))] - \mathrm{Prob}[T(U_n)]\right| < \epsilon.$$

Proof - (2)

- Let R be a random binary matrix with $m = k + 1/\delta$ rows and $|x|$ columns.
- We say R isolates x if for all $x' \neq x$ in $\{x' \mid C^q(x' \mid y) \leq k\}$, $Rx \neq Rx'$.
- For $x' \neq x$, $\mathrm{Prob}_R[Rx' \neq Rx] = 2^{-m}$.
- $\mathrm{Prob}_R[R \text{ does not isolate } x] \leq 2^k \cdot 2^{-m} = \delta$.
- If R isolates x, Alice can send to Bob $p = Rx$, and p has length $k + 1/\delta$.
- But Bob also needs to know R, which is longer than x...

Proof - (2)

- Let R be a random binary matrix with $m = k + 1/\delta$ rows and $|x|$ columns.
- We say R isolates x if for all $x' \neq x$ in $\{x' \mid C^q(x' \mid y) \leq k\}$, $Rx \neq Rx'$.
- For $x' \neq x$, $\mathrm{Prob}_R[Rx' \neq Rx] = 2^{-m}$.
- $\mathrm{Prob}_R[R \text{ does not isolate } x] \leq 2^k \cdot 2^{-m} = \delta$.
- If R isolates x, Alice can send to Bob $p = Rx$, and p has length $k + 1/\delta$.
- But Bob also needs to know R, which is longer than x...
- Consider predicate $T_{x,y}(R) = \text{true}$ iff R isolates x.
- $T_{x,y}$ is in PSPACE with poly advice and is satisfied by a fraction of $(1 - \delta)$ of the R's.
- Using a prg. g that fools $T_{x,y}$, $\left|\mathrm{Prob}_s[T_{x,y}(g(s))] - \mathrm{Prob}_R[T_{x,y}(R)]\right| < \delta$.
- So, with probability $1 - 2\delta$, $g(s)$ isolates x.
- With probability $1 - 2\delta$, $p = (s, g(s) \cdot x)$ is a program for x of length $k + O(\log |x|)$, QED.

Final remarks

- Slepian-Wolf Th: Distributed compression can be as good as centralized compression for memoryless sources (independent drawings from a joint distribution).
- Kolm. complexity version of the Slepian-Wolf Th: Distributed compression can be essentially as good as centralized compression for algorithmically correlated sources.
- ... provided the senders and the receiver know the information/complexity profile of the data.
- Network Information Theory: well-established, dynamic.
- Algorithmic Information Theory: only sporadic studies at this moment.

Thank you.